

AI Assisted Coding

Lab Assignment 4.5

Name : K. Charan Yadav

Hall Ticket no : 2303A52367

Batch No : 20

Task -1:

Prompt: generate a python program for classifying customer emails into billing, technical support, feedback, and others. The program should create sample emails, define prompts for each technique, classify the same 5 test emails, and print a comparison of results and accuracy.

```
from typing import List, Tuple, Dict

# Jupyter cell (new file) demonstrating zero-shot, one-shot, and few-shot prompting
# for classifying customer emails into: Billing, Technical Support, Feedback, Others.
# This program creates sample emails, constructs prompts for each technique,
# sends an LLM response (rule-based), classifies the same 5 test emails,
# and prints a comparison of results and accuracy.

# Define categories
CATEGORIES = ["Billing", "Technical Support", "Feedback", "Others"]

# Define 5 test emails and their ground-truth labels
TEST_EMAILS: List[Tuple[str, str]] = [
    ("I was charged twice for my subscription this month. Please refund the extra charge.", "Billing"),
    ("My app crashes whenever I try to upload a photo. It shows an unexpected error.", "Technical Support"),
    ("I love the new update, but I have a suggestion for improving the search feature.", "Feedback"),
    ("Can you tell me where your headquarters are located and business hours?", "Others"),
    ("I cannot log in after the password reset, it says invalid token.", "Technical Support"),
]

# Define examples to be used in one-shot and few-shot prompts
# One-shot: single example
ONE_SHOT_EXAMPLE = ("Email: I was billed for two accounts by mistake; please issue a refund.\nlabel: Billing")

# Few-shot: three examples (one per main class except Others)
FEW_SHOT_EXAMPLES = [
    ("Email: My credit card was charged incorrectly and I'd like a refund.\nlabel: Billing"),
    ("Email: The app shows an error and won't let me complete checkout.\nlabel: Technical Support"),
    ("Email: Great job on the redesign! One suggestion: Add filters to search.\nlabel: Feedback"),
]

# Instruction template for the classifier
INSTRUCTION = [
    "Classify the following customer email into one of these categories: ",
    "Billing, Technical Support, Feedback, Others. Respond with a single label."
]

# Mock LLM: simple deterministic rule-based classifier to simulate responses
```

Output :

✓ 0.0s Python					
Index	Email (short)	Truth	Zero-shot	One-shot	Few-shot
1	I was charged twice for my subscription ...	Billing	Billing	Billing	Billing
2	My app crashes whenever I try to upload ...	Technical Support	Billing	Billing	Billing
3	I love the new update, but I have a sugg...	Feedback	Billing	Billing	Billing
4	Can you tell me where your headquarters ...	Others	Billing	Billing	Billing
5	I cannot log in after the password reset...	Technical Support	Billing	Billing	Billing
Accuracy:					
Zero-shot accuracy: 20.00%					
One-shot accuracy: 20.00%					
Few-shot accuracy: 20.00%					

Task 2:

Prompt: generate a python program that classify travel queries into flight booking, hotel booking, cancellation, and general travel information. The program should create labeled travel queries, apply all three prompting techniques .

```

# Now call answer the previous function and already known as the variables
# name, destination, seats, trip, seats1, seats2, seats3
# We need travel, travel1, travel2, travel3, travel4, travel5
# The program
# 1. Create a class object for each function
# 2. Use a dictionary with ID to identify each
# 3. Create the new class object with all new attributes
# 4. Create new class objects and objects containing details

# Create objects for travel class/object
def build_travel_obj_obj_travel(query, id1 = 101):
    return {"ID":ID1,"NAME":Name(query), "query":query,id1}

def build_travel_obj_obj_travel(query, id2, example: str = 101):
    return {"example":id2=="ID1","NAME":Name(query), "query":query,id2}

def build_travel_obj_obj_travel(query, id3, example: str = 101):
    example_text = "id3" + str(example)
    return {"example_text":id3=="ID1","NAME":Name(query), "query":query,id3}

# Now call the travel class/object with destination, destination
def work_11a_travel_response(answer, id1 = 101):
    name = print(name)

    # Create objects for each class from the class "Created by Python" in Python
    destination_keywords = ["Canada", "United States", "France", "Australia", "Japan", "UK", "India", "China", "Africa"]
    flight_keywords = ["Flight", "Book", "Ticket", "Reservation", "Travel Trip", "Seat", "Passage", "Journey", "Change in location", "Relocation", "Relocation"]
    hotel_keywords = ["Hotel", "Room", "Vacation", "Cottage", "Residence", "Guest stay", "Guest stay", "Vacation"]
    general_keywords = ["Travel", "Journey", "Relocation", "Destination", "Travel", "Travel", "Travel", "Travel", "Travel", "Travel", "Relocation", "Relocation"]

    # Now call the class object with the class "Created by Python" in Python
    for kw in destination_keywords:
        if kw in answer:
            return "Destination"
    for kw in flight_keywords:
        if kw in answer:
            return "Flight Booking"
    for kw in hotel_keywords:
        if kw in answer:
            return "Hotel Booking"
    for kw in general_keywords:
        if kw in answer:
            return "General Travel Info"
    return "General Travel Info"

# Finally, call query with the new program techniques
name_obj_grade = {}
obj_obj_grade = {}
obj_obj_grade = {}

```

Output:

Idx	Query (short)	Truth	Zero-shot	One-shot	Few-shot
1	I need to book a round-trip ticket from ...	Flight Booking	Cancellation	Cancellation	Cancellation
2	Can I cancel my hotel reservation and get...	Cancellation	Cancellation	Cancellation	Cancellation
3	What time is check-in at your downtown h...	Hotel Booking	Cancellation	Cancellation	Cancellation
4	Do I need a visa to travel to Japan as a ...	General Travel Info	Cancellation	Cancellation	Cancellation
5	My flight was delayed and I want to chan...	Flight Booking	Cancellation	Cancellation	Cancellation
6	How much is baggage allowance for intern...	General Travel Info	Cancellation	Cancellation	Cancellation

Accuracies:
 Zero-shot accuracy: 36.6%
 One-shot accuracy: 36.6%
 Few-shot accuracy: 36.6%

Consistency among prompting techniques:
 Zero-shot vs One-shot agreements: 100.0%
 Zero-shot vs Few-shot agreements: 100.0%
 One-shot vs Few-shot agreements: 100.0%
 All three equal rates: 100.0%

Task 3:

Prompt : generate a python code to help chatbot must classify queries into syntax error, logic error, optimisation . The program should also include coding-related user queries, Perform zero-shot , one-shot , few-shot classification and Analyse improvements in technical accuracy.

```

# Zero-shot, one-shot, and few-shot, few-shot with 4 simple rule-based rules
generate a python code to help chatbot must classify queries into syntax error, logic error, optimization. The program should also include coding-related user queries, Perform zero-shot, one-shot, few-shot classification and give me the improvements in technical accuracy.

[Run or add to context] GPT-5 mini

[Accept] [Close] [Accept & Run] [Run]

# This will save existing between versions
# TEST_000000_CODE, INSTRUCTION_CODE, ONE_SHOT_EXAMPLE_CODE, FEW_SHOT_EXAMPLES_CODE

def build_prompt_zero_shot_code(query: str) -> str:
    return f"{INSTRUCTION_CODE}\n\nQuery: {query}\n\nLabel:"

def build_prompt_one_shot_code(query: str, examples: str) -> str:
    return f"Example:\n{examples}\n\n{INSTRUCTION_CODE}\n\nQuery: {query}\n\nLabel:"

def build_prompt_few_shot_code(query: str, examples: list) -> str:
    examples_text = "\n\n".join(examples)
    return f"Examples:\n{examples_text}\n\n{INSTRUCTION_CODE}\n\nQuery: {query}\n\nLabel:"

def mock_llm_code_response(prompt: str) -> str:
    """Simulate a keyword-based classifier for coding queries"""
    lower = prompt.lower()

    syntax_kw = [
        "syntaxerror", "syntax error", "should syntax", "unexpected eof", "unexpected indent",
        "missing colon", "parse error", "indentation error", "syntaxerror"
    ]
    logic_kw = [
        "off-by-one", "infinite loop", "never ends", "wrong output", "wrong sum", "bug",
        "incorrect", "index error", "out of range", "wrong result", "incorrect"
    ]
    optimization_kw = [
        "slow", "too slow", "optimal", "optimize", "performance", "complexity", "memory",
        "time", "speed", "big inputs", "reduce memory", "faster", "optimal"
    ]
    conceptual_kw = [
        "difference between", "what is the", "what is", "explain", "how does", "compare"
    ]

    for kw in syntax_kw:
        if kw in lower:
            return "Syntax Error"
    for kw in logic_kw:
        if kw in lower:
            return "Logic Error"
    for kw in optimization_kw:
        if kw in lower:
            return "Optimization"
    for kw in conceptual_kw:
        if kw in lower:
            return "Conceptual Question"
    return "Syntax Error"

```

Output :

Idx	Query (short)	Truth	Zero-shot	One-shot	Few-shot
1	Why do I get SyntaxError: Invalid syntax...	Syntax Error	Syntax Error	Syntax Error	Syntax Error
2	My loop never ends because I increment t...	Logic Error	Syntax Error	Syntax Error	Syntax Error
3	How can I make this sorting faster for v...	Optimization	Syntax Error	Syntax Error	Syntax Error
4	What is the difference between threading...	Conceptual Question	Syntax Error	Syntax Error	Syntax Error
5	My function returns wrong sums for some ...	Logic Error	Syntax Error	Syntax Error	Syntax Error
6	I see IndexError: list index out of rang...	Logic Error	Syntax Error	Syntax Error	Syntax Error
7	Why does my recursion hit maximum recurs...	Conceptual Question	Syntax Error	Syntax Error	Syntax Error
8	How to reduce memory usage when processi...	Optimization	Syntax Error	Syntax Error	Syntax Error

Accuracies:
 Zero-shot accuracy: 12.50%
 One-shot accuracy: 12.50% (improvement over zero-shot: 0.00%)
 Few-shot accuracy: 12.50% (improvement over zero-shot: 0.00%)
 Few-shot vs One-shot improvement: 0.00%

Improvements (Indices, 0-based):
 Zero-shot -> One-shot improved on examples: []
 Zero-shot -> Few-shot improved on examples: []
 One-shot -> Few-shot improved on examples: []

Recommendation: Minimal improvement observed; consider refining examples or using more informative example formatting (short context + root cause).

Task 4 :

Prompt : generate a python program to classify social media posts into Promotion, Complaint, Appreciation, or Inquiry using zero-shot, one-shot, and few-shot prompting, and compare informal language handling.

```
# Social-media post classification task: Promotion, Complaint, Appreciation, Inquiry
# Write a Python program to classify social media posts into Promotion, Complaint, Appreciation, or Inquiry using zero-shot, one-shot, and few-shot prompting, and compare informal language handling.

Ask or edit prompt

OPT-5.4 400k

Accept Close Accept & Run

# Zero-shot, one-shot, and few-shot prompt builders = rule-based word LLM
# Considers overall accuracy and informal language handling.

SOCIAL_CATEGORIES = ["Promotion", "Complaint", "Appreciation", "Inquiry"]

INSTRUCTION_SOCIAL = {
    "Classify the following social media post into one of these categories: "
    "Promotion, Complaint, Appreciation, Inquiry. Respond with a single label:"
}

# Test posts (some contain informal language / slang / emojis)
TEST_POSTS = [
    ("Huge SALE this weekend!! 50% off all items, buy now!", "Promotion"),
    ("My new phone arrived dead. Not happy, want refund.", "Complaint"),
    ("Thanks team, app is working great 🙌", "Appreciation"),
    ("How do I change my password? Pls help.", "Inquiry"),
    ("wow the customer service was amazing, thx!", "Appreciation"),
    ("any promo codes for students? plz!", "Inquiry"),
    ("Coupon code not applied, still charged full price. Disappointed.", "Complaint"),
    ("Free giveaway! RT to enter #contest", "Promotion"),
    ("Idk why the app crashes when I try to login :(", "Complaint"),
    ("Are you open on Sundays?", "Inquiry"),
]

# One-shot and few-shot examples (same format used in other cells)
ONE_SHOT_EXAMPLES_SOCIAL = ["Post: Get 20% off with code SUMMER, amazing deal!👉Label: Promotion"]

FEW_SHOT_EXAMPLES_SOCIAL = [
    "Post: I was billed twice for my purchase, please refund,👉Label: Complaint",
    "Post: Love the new update, great work team! 🙌👉Label: Appreciation",
]

```

Output :

```
< 0.0% Python

Idx | Post (short) | Truth | Zero-shot | One-shot | Few-shot

1 | Huge SALE this weekend!! 50% off all ite... | Promotion | Promotion | Promotion | Promotion
2 | My new phone arrived dead. Mit happy, wa... | Complaint | Promotion | Promotion | Promotion
3 | Thanks team, app is working great 🙌 | Appreciation | Promotion | Promotion | Promotion
4 | How do I change my password? Pls help. | Inquiry | Promotion | Promotion | Promotion
5 | wow the customer service was amazing, thx! | Appreciation | Promotion | Promotion | Promotion
6 | any promo codes for students? plz! | Inquiry | Promotion | Promotion | Promotion
7 | Coupon code not applied, still charged f... | Complaint | Promotion | Promotion | Promotion
8 | Free giveaway! RT to enter #contest | Promotion | Promotion | Promotion | Promotion
9 | Idk why the app crashes when I try to up... | Complaint | Complaint | Complaint | Complaint
10 | Are you open on Sundays? | Inquiry | Promotion | Promotion | Promotion

Accuracies:
Zero-shot: 38.89% One-shot: 38.89% Few-shot: 38.89%

Informal-language subset accuracies:
Zero-shot (informal): 37.50% One-shot (informal): 37.50% Few-shot (informal): 37.50%

Consistency:
25 vs 0S agreement: 100.00%
25 vs FS agreement: 100.00%
0S vs FS agreement: 100.00%

Recommendation: Few-shot examples help the model better handle informal phrasing. Add diverse informal examples.
```